

Accelerated Portfolio Optimization with Conditional Value-at-Risk Constraints using a Cutting-Plane Method*

GEORG HOFMANN[†]

Abstract. Financial portfolios are often optimized for maximum profit while subject to a constraint formulated in terms of the Conditional Value-at-Risk (CVaR). This amounts to solving a linear problem. However, in its original formulation this linear problem has a very large number of linear constraints, too many to be enforced in practice. In the literature this is addressed by a reformulation of the problem using so-called dummy variables. This reduces the large number of constraints in the original linear problem at the cost of increasing the number of variables. In the context of reinsurance portfolio optimization we observe that the increase in variable count can lead to situations where solving the reformulated problem takes a long time. Therefore we suggest a different approach. We solve the original linear problem with cutting-plane method: The proposed algorithm starts with the solution of a relaxed problem and then iteratively adds cuts until the solution is approximated within a preset threshold. This is a new approach. For a reinsurance case study we show that a significant reduction of necessary computer resources can be achieved.

Key words. Optimization, financial portfolio, linear programming, conditional value-at-risk constraint, cutting plane

AMS subject classifications.

52A40 Inequalities and extremum problems

90C05 Linear programming

90C90 Applications of mathematical programming

90B50 Management decision making, including multiple objectives

1. Introduction. In the industry, risk management of financial portfolios is commonly approached with a Monte Carlo simulation. The model typically consists of a $J \times n$ -matrix Y . Its J rows represent scenarios, i.e. outcomes of a simulation that are considered to be equally probable. Its n columns represent different instruments that make up the portfolio. The entries of Y represent the value of an instrument for a particular outcome. So the column means of Y yield the expected value of the instruments. The row sums of Y yield the portfolio value for each scenario. We call this the *outcome vector*. In the context of reinsurance portfolios the value of instruments can become negative due to catastrophic loss simulated in the scenarios. However the expected instrument value is typically positive due to premium collected. We refer to the matrix Y as the *scenario matrix*.

In this article we represent the risk of a portfolio by the Conditional Value-at-Risk (CVaR), also called the Tail Value-at-Risk (TVaR).¹ In the model above the CVaR at a return period ρ can be calculated as follows, provided that the number of scenarios J is a multiple of the return period ρ : Let y be the row sums of Y . Let y' be a reordering of y in a way that the components y' are increasing. Define the vector r by setting

$$r_j = \begin{cases} -\frac{\rho}{J} & \text{for } i = 1, 2, \dots, \frac{J}{\rho} \\ 0 & \text{for } i = \frac{J}{\rho} + 1, \frac{J}{\rho} + 2, \dots, J. \end{cases}$$

*The US patent [1] uses methods from this article.

[†]The Research for this paper is supported by Validus Research Inc., Waterloo Ontario, Canada.

¹For a valuable discussion of risk metrics commonly used in this context, see the introduction of [2]. Of particular interest is the comparison with the most common risk metric, the Value-at-Risk (VaR) which is simply a percentile of the loss distribution.

for every $j = 1, 2, \dots, J$. Then the desired CVaR is given by the matrix product $r^T y'$. An equivalent way of defining the CVaR of y is the following:

$$(1.1) \quad \mu_r(y) = \max_{\pi \in S_J} r^T P_\pi y,$$

where S_J is the set of all permutations on the set $\{1, 2, \dots, J\}$ and P_π is the permutation matrix associated with the permutation π . We refer to r as the *risk vector* associated with the *risk metric* μ_r .

In order to model change in the portfolio composition an n -dimensional vector x is used that is subject to the constraints

$$(1.2) \quad \underline{x} \leq x \leq \bar{x},$$

where \underline{x} and \bar{x} are both n -dimensional vectors. The inequality (1.2) is to be read component by component. In other words, if x_i denotes the i th component of x then we enforce the following inequality for every $i = 1, 2, \dots, n$:

$$\underline{x}_i \leq x_i \leq \bar{x}_i.$$

We refer to x as a *position vector* and to (1.2) as the *position constraint*. For a given position vector x the altered scenario matrix is obtained by multiplying the columns of Y with the components of x . The altered outcome vector is simply the product Yx .

Suppose the risk of the altered portfolio to be bounded by a number R , in other words the inequality

$$\mu_r(Yx) \leq R$$

is to be enforced. An equivalent way of formulating this constraint is

$$r P_\pi Y x \leq R \quad \text{for every permutation } \pi \in S_J.$$

This is a system of $J!$ linear inequalities. The number of inequalities could be reduced to account for the fact that r contains zeros, but it is generally still very large. In practice this prevents the problem from being passed to a standard algorithm in its original formulation.

Recently a practical solution has been presented in [3] and refined in [4] and [2]. It is based on a reformulation of the problem that increases the number of variables from n to $n + J + 1$ but decreases the number of constraints to $2J + 2n + 1$. In many cases this makes the problem accessible to standard algorithms for linear problems. It allows to process 'over one hundred instruments and one thousand scenarios'. (See [2], Introduction, page 2.)

However, as the number of scenarios J increases, these algorithms may still take a long time to complete or fail due to a overly high demand for computer resources. In the reinsurance industry a simulation with 1 million scenarios is not an exception, neither is the usage of 10 thousand instruments.

We propose a different approach to solving the linear problem at hand. We address the original problem using a cutting-plan method. The iterative algorithm starts with only the position constraints enforced. Then, step by step, relevant constraints, so-called cuts, are enforced. The algorithm terminates once the set of constraints is large

enough to force the observed risk R^* to be sufficiently close to R . To be more precise, a *risk error tolerance* δ can be specified. The termination condition is then given by:

$$\left| \frac{R^* - R}{R} \right| \leq \delta$$

In a case study we mimic typical reinsurance portfolio scenarios of different sizes. We provide the number of iterations when our algorithm is applied and also give a comparison of run times when the algorithm proposed by [3] and our algorithm are applied. We were able to run our algorithm for one million scenarios and 10 thousand instruments, and this data size does not form an upper bound.

While this case study is motivated from practice in the reinsurance industry, the algorithm we propose is not at all limited to this industry. It is applicable to optimization of any financial portfolio. In Section 6 we list some interesting linear constraints that an extension of the proposed algorithm can handle. Applications beyond the financial portfolios could be found in other areas of risk management.

2. The Linear Problem. Throughout the remainder of the article assume that the number of instruments n and the number of scenarios J are given. Let x be a position vector subject to the position constraint

$$(2.1) \quad \underline{x} \leq x \leq \bar{x}.$$

Fix a scenario matrix Y . Let r be a J -dimensional vector with non-positive components. The example of a TVaR risk vector from the introduction is instructive. But other choices are possible, in particular a weighted combination of different TVaR risk vectors is admissible. In addition to constraint (2.1) we enforce

$$(2.2) \quad rP_\pi Yx \leq R \quad \text{for every permutation } \pi \in S_J.$$

Let p be an n -dimensional vector. It is helpful to think of p as the vector column sums of Y . This way, the $p^T x$ represents the expected altered portfolio value. But other choices of p are possible.

The linear problem at hand is to

$$(2.3) \quad \text{Maximize } p^T x \text{ subject to the constraints (2.1) and (2.2).}$$

Throughout the remainder of the article we will refer to this as the *original linear problem*. If there is at least one vector x that satisfies the constraints of the problem, then the problem has a solution since x is constrained to a compact domain. Typically the n dimensional vector of ones lies within the constraints, as it represents the unaltered portfolio, which usually satisfies the risk constraints.

3. The Algorithm. In practice the number of constraints in (2.2) is too large for all of them to be enforced. We propose the following algorithm to add only constraints that are relevant for the solution. This approach is similar to the cutting-plane method.

The following inputs are required. We use the terminology and the notation of previous sections. The following are the input parameters:

Y	Scenario matrix
p	Profit vector
\bar{x}	Upper position constraint vector
\underline{x}	Lower position constraint vector
R	Target risk
δ	Risk error tolerance

The following steps describe the algorithm.

1. *Formulate the relaxed problem:* Initiate the set of constraints C as the constraints defined in (2.1).
2. *Solve the current problem:* Maximize $p^T x$ subject to the constraints in C . Denote the solution by x^* .
3. *If the achieved risk R^* is close enough to the target risk R , go to Step 6:* Set

$$R^* = \mu_r(Yx^*).$$

If $|R^* - R| \leq \delta R$ then jump to Step 6.

4. *Add a constraint to the problem:* Let π be a permutation such that $P_\pi Yx^*$ is in increasing order. Add the constraint

$$rP_\pi Yx \leq R$$

to the set C .

5. *Return to Step 2.*
6. *Optional verification of the solution to the relaxed problem:* By solving the dual problem to the one formulated in Step 2, it can be verified that an optimal solution has been obtained.
7. *Algorithm output:* The output of the algorithm is the solution x^* , the achieved risk R^* and the obtained profit $s = p^T x^*$.

This is an iterative algorithm and its output is a numerical approximation of the theoretical solution. In the next section we describe the nature of this approximation and how its error can be quantified. In the section following that we provide a case study that includes the number of iterations observed for actual data. In reality the approximation error is comparable to rounding errors that are typical for any numerical solution.

4. Convergence. In this section we give a precise description of how the solution achieved from the iterative algorithm approximates the solution of the original linear problem (2.3). Let D be the set of all numbers R' for which the following set is not empty:

$$\{x' \in [\underline{x}, \bar{x}] : \mu_r(Yx') \leq R'\}$$

Note that the target risk R should be in this set, otherwise constraints (2.1) and (2.2) can't be satisfied. For every $R' \in D$ set

$$(4.1) \quad f(R') = \max_{\substack{\underline{x} \leq x' \leq \bar{x} \\ \mu_r(Yx') \leq R'}} p^T x'.$$

This is well-defined, since we are taking the maximum of a continuous function over a compact set. So, this assignment defines a function from D to \mathbb{R} . The function f is called the *efficient frontier*. Note that $f(R)$ is precisely the maximum profit that is to be determined in the original problem (2.3).

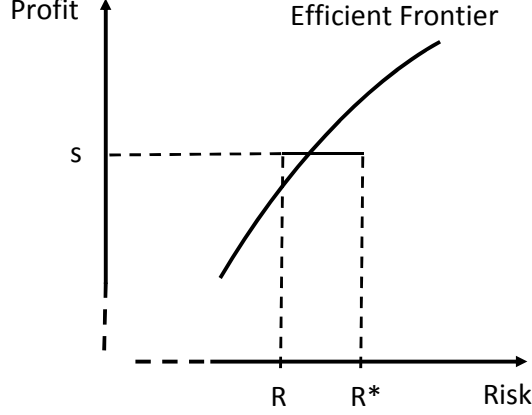


FIG. 1. The line segment from (R, s) to (R^*, s) intersects the efficient frontier. This guarantees that the algorithm approximates a point on the frontier.

Now consider the algorithm again with the target risk R . Assume that a solution to the original problem exists. Then solutions exists to any of the relaxations of the problem occurring in the algorithm. Suppose the algorithm terminates with the achieved risk R^* and the obtained profit s . Since these values resulted from a relaxed problem we have $s \geq f(R)$. On the other hand, due to the definition of f , we have $s \leq f(R^*)$.

This means that there is a number R' with $R \leq R' \leq R^*$ such that $f(R') = s$. In other words, there is a frontier point between the points (R, s) and (R^*, s) in the efficient frontier diagram. See Figure 1. This follows from the Intermediate Value Theorem. A proof that f is actually continuous is provided in Appendix A.

Since the distance between R and R^* can be controlled by specifying δ , the distance between the output point (R^*, s) and a point on the frontier is also controlled by δ .

5. A Case Study. In this section, the application of the two methods is compared:

- (A) The method proposed in [3], which relies on a reformulation of the underlying linear problem.
- (B) The method proposed in this article, which uses a cutting-plane approach to address the underlying linear problem.

A few combinations of values for n and J are selected and an example of Y is generated to allow a comparison on actual data. The scenario matrix Y created randomly with a typical reinsurance portfolio in mind. However it is important to emphasize that it is not based on any true data and that it should not be used to represent in any way a real portfolio.

The matrix Y is created in the following way. An $f \times n$ matrix L is created where $f = 100$ in this case study. Its entries are randomly generated between 0 and 1 according to a uniform distribution. A $J \times f$ matrix F is created. Each entry is randomly generated. The underlying distribution is derived from a log normal by an

J	n	Iterations		Variables		Constraints	
		(A)	(B)	(A)	(B)	(A)	(B)
1,000	100	1	4	1,101	100	2,201	204
10,000	200	1	14	10,201	200	20,401	414
100,000	500	1	58	100,501	500	201,001	1,058
1,000,000	1,000	1	223	1,001,001	1,000	2,002,001	2,223

TABLE 1
Comparison of Approach (A) and (B)

$J \backslash n$	100	200	500	1,000
1,000	13	13	11	8
2,000	22	26	19	15
5,000	43	48	43	41
10,000	96	90	115	97

TABLE 2
Acceleration of run time when moving from Approach (A) to Approach (B).

affine transformation such that defined as followed: If N is distributed according to a standard normal distribution, then the entries of F are sampled from the distribution of the transformed variable $2 - e^N$. Its support is $(-\infty, 2]$ and its expected value is $2 - \sqrt{e} \approx 0.351$. The matrix Y is finally computed as the product FL . In this way each instrument is a random linear combination of f factors. This introduces a correlation between instruments that we consider typical for a reinsurance portfolio.

In Appendix B, code in the language R is provided to show how the scenario matrix Y is created and how the algorithm is implemented. For each of the two methods Table 1 provides information about the number constraints and variables in the linear problem to be solved. Since method (B) is iterative, several linear problems have to be solved in the course of the algorithm, as the number of constraints, we report the number of constraints in the last iteration. Table 2 provides a comparison in run times. The factor report is the run time of method (A) divided by method (B). For both tables the CVaR at a return period of 100 was used.

6. Extensions of the Algorithm. As discussed earlier, the application of the algorithm is not limited to the reinsurance case investigated in the previous section. It can be used to optimize financial portfolios in general. Further applications in areas of risk management are likely.

The algorithm can also be extended to handle further linear constraints. These include the constraints described in [2]: In that article they are referred to as

1. Transaction Cost Balance Constraints ([2] 7.3)
2. Value Constraints ([2] 7.4)
3. Liquidity Constraints ([2] 7.5)

Note that the constraints referred to as *Bounds on Positions* in [2] 7.5, are effectively the constraints we cover with (2.1).

In practice it is often desirable to calculate segments of the efficient frontier describing the trade-off between risk and profit. The proposed algorithm can be used to accomplish this by stepping through a range of risk values. For each risk value the optimal profit value is calculated. This method can provide a desirable range of points on the frontier.

7. Conclusion. We present a novel approach to solving an existing problem in practice. In a case study that is geared towards reinsurance business, we show that a significant gain in performance and a reduction in computer resources can be achieved using our method. We prove that the proposed algorithm approximates an efficient frontier point within a customizable accuracy. With the optional validation step in the algorithm, it can be guaranteed that an optimal solution has been captured.

Appendix A. Proposition and Proof.

Let X be a non-empty compact convex subset of \mathbb{R}^n . Let

$$\mu : X \rightarrow \mathbb{R}$$

be a continuous, convex function. Note that the image $\mu(X)$ is compact so we can set

$$d = \min(\mu(X)).$$

Let

$$p : \mathbb{R}^n \rightarrow \mathbb{R}$$

be a concave continuous function and define the function

$$(A.1) \quad f : [d, \infty) \rightarrow \mathbb{R}, \quad f(R) = \max_{x \in \mu^{-1}((-\infty, R])} p(x)$$

$$(A.2) \quad = \max_{\substack{x \in X \\ \mu(x) \leq R}} p(x)$$

It is well-defined, since $\mu^{-1}((-\infty, R])$ is compact and p is continuous. Note that f is an increasing function.

PROPOSITION A.1. *The function f is continuous.*

Proof. First we prove that f is concave. That implies that f is continuous on any open interval. Once this is established, the only way that continuity of f can fail is that it jumps at d . In a second step we will show that this is not possible.

For the proof of concavity let R_1 and $R_2 \in D$. Now let x_1 and $x_2 \in X$ such that

$$(A.3) \quad x_i \in \mu^{-1}((-\infty, R_i]) \quad \text{and} \quad f(R_i) = p(x_i)$$

for each $i = 1, 2$. Now let $t \in [0, 1]$. Then

$$\begin{aligned} \mu(tx_1 + (1-t)x_2) &\leq t\mu(x_1) + (1-t)\mu(x_2) && \text{since } \mu \text{ is convex} \\ &\leq tR_1 + (1-t)R_2 && \text{according to (A.3),} \end{aligned}$$

in other words

$$(A.4) \quad tx_1 + (1-t)x_2 \in \mu^{-1}((-\infty, tR_1 + (1-t)R_2]).$$

In turn, this implies

$$\begin{aligned} tf(R_1) + (1-t)f(R_2) &= tp(x_1) + (1-t)p(x_2) && \text{according to (A.3)} \\ &\leq p(tx_1 + (1-t)x_2) && \text{since } p \text{ is concave} \\ &\leq \max_{x \in \mu^{-1}((-\infty, tR_1 + (1-t)R_2])} p(x) && \text{due to (A.4)} \\ &= f(tR_1 + (1-t)R_2). \end{aligned}$$

This proves that f is concave.

Now let (R_i) be a sequence in (d, ∞) with $\lim_{i \rightarrow \infty} R_i = d$. We will prove

$$(A.5) \quad \lim_{i \rightarrow \infty} f(R_i) = f(d)$$

in order to see that f is continuous at d . Without loss of generality we may assume that (R_i) is decreasing. Since f is an increasing function the sequence $(f(R_i))$ is also decreasing. Since it is bounded from below by $\min p(X)$ it is convergent. By the definition of f there is an $x_i \in X$ such that

$$\mu(x_i) \leq R_i \quad \text{and} \quad p(x_i) = f(R_i)$$

for every $i \in \mathbb{N}$. In this way we obtain a sequence (x_i) . Since X is compact, there is a subsequence (x_{i_j}) that converges to an $x \in X$. We observe that

$$\begin{aligned} f(d) &\leq \lim_{j \rightarrow \infty} f(R_{i_j}) && \text{since } f \text{ is increasing} \\ &= \lim_{j \rightarrow \infty} p(x_{i_j}) = p(x) \\ &\leq \max_{x' \in X, \mu(x') \leq d} p(x') = f(d) \end{aligned}$$

This implies $\lim_{j \rightarrow \infty} f(R_{i_j}) = f(d)$ and thus (A.5). \square

This proposition can be applied to the case

$$\mu(x) = \mu_r(Yx)$$

for the function μ_r defined in (1.1). For that it should be noted that μ_r is continuous, since it is the maximum of a finite number of continuous functions.

Appendix B. R Code.

The code below runs in R version 3.0.2 once the package `lpSolve` is loaded. This package is based on `lp_solve` 5.5.

```
# This code is part of the article "Accelerated Portfolio Optimization
# with Conditional Value-at-Risk Constraints using a Cutting-Plane Method
library(lpSolve)

### Preset constants.
num.scenarios <- 1E4
num.instruments <- 1000

### Functions
GetRiskMetricVector <- function(return.period){
  rp.scenarios <- num.scenarios / return.period
  return(c(rep(-1 / rp.scenarios, rp.scenarios),
            rep(0, num.scenarios - rp.scenarios)))
}

### Main Code
# The variables F, L, Y, p, r, delta are explained in the article.
f=100
F <- matrix(2 - rlnorm(n=num.scenarios * f), nrow=num.scenarios)
```



```

L <- matrix(runif(num.instruments * num.factors), nrow=f)
Y <- F %%% L
delta <- 1E-6
# As a risk metric we use the average of the CVaR 100 and the CVar 1000
r <- 0.5 * GetRiskMetricVector(100) + 0.5 * GetRiskMetricVector(1000)
A <- rbind(diag(nrow=num.instruments), diag(nrow=num.instruments))
b <- c(rep(1.5, num.instruments), rep(0.5, num.instruments))
p <- colSums(Y) / num.scenarios
constr.vec <- c(rep("<=", num.instruments), rep(">=", num.instruments))
# We set the risk constraint to the level of risk in the original portfolio:
scenario.outcome <- rowSums(Y)
scenario.outcome <- scenario.outcome[order(scenario.outcome)]
risk.constraint <- r %%% scenario.outcome
repeat{
  lp.sol <- lp(direction="max", objective.in=p, const.mat=A,
               const.dir=constr.vec, const.rhs=b)

  if(lp.sol$status>0)stop("Lp solve error: ", lp.sol$status)
  scenario.outcome <- Y %%% lp.sol$solution
  new.order <- order(scenario.outcome)
  inv.new.order <- invPerm(new.order)
  r.reordered <- r[inv.new.order]
  achieved.risk <- as.vector(r.reordered %%% scenario.outcome)
  if(achieved.risk - risk.constraint < delta) break
  A <- rbind(A, as.matrix(t(r.reordered) %%% Y))
  b <- c(b, risk.constraint)
  constr.vec <- c(constr.vec, "<=")
}
cat("Risk constraint:", risk.constraint, "\r\n")
cat("Achieved risk:", achieved.risk, "\r\n")
cat("Profit:", lp.sol$objval, "\r\n")

```

REFERENCES

- [1] GEORG W. HOFMANN, *Portfolio optimization and evaluation tool*, July 2014. US Patent application number 14336632.
- [2] PAVLO KROKHMAL, JONAS PALMQUIST, AND STANISLAV URYASEV, *Portfolio optimization with conditional value-at-risk objective and constraints*, Journal of Risk, 4 (2002), pp. 11–27.
- [3] R. TYRRELL ROCKAFELLAR AND STANISLAV URYASEV, *Optimization of conditional value-at-risk*, Journal of Risk, 2 (2000), pp. 21–41.
- [4] STANISLAV URYASEV, *Conditional value-at-risk: Optimization algorithms and applications*, Financial Engineering News, 14 (2000), pp. 1–5.